EXHIBIT 16

CISCO SYSTEMS

Document Number	ENG-103148
Revision	8
Author	Shyam Kaluve, Christian Benvenuti, François Tallet
Project Manager	Al Annamalai

IOS Rapid STP(802.1w) and Multiple STP (802.1s) Functional Spec

This document describes functional specification of IEEE Rapid Spanning Tree Protocol (IEEE 802.1w standard) and IEEE Multiple Spanning Tree Protocol (IEEE 802.1s) implementation in Cisco LAN Switches running IOS with backward compatibility to Cisco PVST+ implementation.

Reviewers

Department	Name	AcceptanceDate
Software Engineering	Bill Rainey, Norman Finn	
Marketing	David Powers	

Modification History

Rev.	Date	Originator	Comment
1	05/16/2001	Christian	Initial draft.
2	08/21/2001	François	Updated the mst configuration mode after parser-police consultation.
3	09/04/2001	'	Added the link-type command that was missing
4	09/11/2001	'	IST is now referred as instance 0, changed show commands description to a basic subset.
5	10/9/2001	'	Updated "portfast default" and "loopguard default" after parser-police posting. Added spanning-tree mode that was missing
6	10/30/2001	'	Added bpduguard and bpdufilter commands.
7	03/01/2002	'	New command (pvlan synchro) + show commands for stp and mst modified.
8	04/10/2002	'	Updated the show commands and added clear spanning-tree counters command.

Definitions

This section defines words, acronyms, and actions which may not be readily understood.

STP	Spanning Tree Protocol
RSTP	Rapid STP (IEEE 802.1w)
PVST+	Per-VLAN Spanning Tree (IEEE 802.1Q compatible)
MISTP	Multi-Instance STP (Cisco Proprietary)
CST	Common Spanning Tree (also VLAN 1 spanning tree in PVST+)
SSTP	Shared Spanning Tree Protocol (not same as MISTP)
SSTP BPDU	BPDU with Cisco STP SNAP encapsulation and Cisco STP destination address
MSTP	Multiple Spanning Tree Protocol (IEEE 802.1s)
CISTP	Common and Internal Spanning Tree Protocol (IEEE 802.1s)
SST	Single Spanning Tree (802.1D/1Q/1w compatible)
MST	Multiple Spanning Tree (802.1s)
I-BPDU	CISTP BPDU
M-BPDU	MSTP BPDU
ISL	Inter Switch Link

Table Of Contents

1.0 Introduction

This document describes the platform specific functional aspects of RSTP [1] and MST [2] protocols. The architectural requirements for a Cisco specific implementation are described in [3] and [4] provides a detailed functional description of both protocols. This document requires a moderate understanding of the protocol functionality, so the reader is urged to take a detailed look at [4] before proceeding further.

As of this writing, this document is applicable to Cosmos and Catalyst 6000 hardware platforms only. However the intention is to extend this implementation to all the other branches of IOS and other hardware platforms where Bridge Group STP implementation or VLAN Switch Shim based PVST+ is in use.

This document introduces all the new spanning tree commands added with the bubble release. Commands that are specific to MST are presented in different chapters from the ones that simply enhance the current spanning-tree implementation.

2.0 RSTP (IEEE 802.1w)

Cisco IOS bridging can be configured to run many flavors of STP. The core Spanning Tree module will be enhanced to provide RSTP as a new protocol type for a given spanning tree instance. However, on switches with integrated L3 routing and switching capability, (owing to reasons that cannot be explained here) the spanning tree instances for the bridge groups containing VLAN interfaces were restricted to run the VLAN Bridge STP. RSTP protocol implementation can be tweaked to use the VLAN Bridge STP encapsulation, but there are no significant benefits in introducing such, yet another, non-standard STP protocol. Therefore RSTP protocol type will not be available for bridge group STP instances on platforms integrating L2 VLAN switching.

<< We will implement PVRST− in a later phase. This will be a different spanning tree mode: all instances will be running RSTP in PVRST− mode. This solution, simple and efficient, will allow users to quickly benefit from 802.1w performance with no change in their design. It should inter operate easily with external routers. On the other hand, it will not provide the extended scalability of MST.>>

3.0 MST (IEEE 802.1s)

Cisco IOS already contains a *VLAN Switch Shim* based PVST+ implementation. MST and PVST+ implementations will co-exist and utilize the functionality provided by the *VLAN Switch Shim* (Efforts to implement our proprietary MISTP[5] has been stopped as MST will provide similar functionality). Though, only one of these flavors can be active at any time. The shim will have the responsibility of ensuring that, as well as changing from one mode to another. The CLI described in section 4 aligns well with the existing CLI for PVST+ and the CLI proposed for CatOS MST implementation [6].

Unlike MISTP, that utilizes VTP to propagate this information, the first implementation of MST assume a VLAN to instance mapping hardcoded on each and every machine running the protocol. This VLAN to instance mapping, that we call the MST configuration (or region configuration) has to be specified by the user to be identical on the whole MST region. At that stage, this configuration will be possible via the CLI and SNMP.

Any change in the region configuration may bring out a drastic change in the network topology, and will certainly disrupt connectivity, at least temporarily. Therefore there is a significant advantage in buffering the region configuration commands, and apply them at once, much like the VLAN database configuration does.

In order to satisfy these goals, the CLI to configure the MST instance to VLAN mapping shall be made part of the spanning tree CLI, independently of the VLAN database CLI. In other words, instance is no more an attribute of VLAN, and rather, VLAN range becomes an attribute of the spanning tree instance. Also, a new submode under the global configuration mode is introduced for the region configuration commands. All the configuration commands entered in the submode will only take effect at once upon exiting the submode.

4.0 CLI

To address the new MST features a new set of commands will be added to the CLI responsible for the spanning tree management. The new commands are needed in configuration mode for both the global configuration and the interface submode, as well as for the show commands. The new keyword *mst* is added where necessary to differentiate the new MST commands from the old ones.

```
(config)#spanning-tree ?
  backbonefast  Enable BackboneFast Feature
  etherchannel  Spanning tree etherchannel specific configuration
  extend        Spanning Tree 802.1t extensions
  loopguard     Spanning tree loopguard options
  mode          Spanning tree operating mode
  mst           Multiple spanning tree configuration
  pathcost      Spanning tree pathcost options
  portfast      Spanning tree portfast options
  uplinkfast    Enable UplinkFast Feature
  vlan          VLAN Switch Spanning Tree

(config)#spanning-tree mst ?
  <0-15>          MST instance id
  configuration   Enter MST configuration submode
  forward-time    Set the forward delay for the spanning tree
  hello-time      Set the hello interval for the spanning tree
  max-age         Set the max age interval for the spanning tree
  max-hops        Set the max hops value for the spanning tree
```

In PVST, by definition, all the spanning tree instances are independent. This is not the case any more with MST, where two kinds of spanning tree instances coexist:

- a unique CIST instance
- several MST instances

Only the CIST instance is able to send and receive BPDUs but MST instances add their information into these BPDUs to interact with neighboring switches and compute their final topology. Because of this, all the usual spanning tree parameters related to BPDU transmission can only be configured on the CIST instance (this comprise the hello-time, forward-time, max-age and the new parameter max-hops) while the parameters related to the spanning tree topology are available on the CIST instance and the MST instances as well (this concern mainly the bridge priority, port instance cost and port instance priority).

As the CIST timers have an impact on all instances, we decided to link their configuration to the MST protocol as a whole, not to a specific instance. Thus the commands:

```
(config)# spanning-tree mst ?
  forward-time    Set the forward delay for the spanning tree
  hello-time      Set the hello interval for the spanning tree
  max-age         Set the max age interval for the spanning tree
  max-hops        Set the max hops value for the spanning tree
```

The CIST instance can then be considered as a regular MST instance as far as the configuration of the remaining parameters is concerned. From now, we will refer to the CIST instance as MST instance 0.

The show commands also need to be updated to take into account the new flags and information introduced with the new protocol, like for instance the port role. For this reason the *mst* keyword is introduced in the show spanning-tree command:

```
#show spanning-tree ?
  <1-255>           Bridge Group number
  active            Report on active interfaces only
  backbonefast      Show spanning tree backbonefast status
  blockedports      Show blocked ports
```

```
  bridge              Status and configuration of this bridge
  detail              Detailed information
  inconsistentports   Show inconsistent ports
  interface           Spanning Tree interface status and configuration
  mst                 Multiple Spanning Trees
  pathcost            Show Spanning pathcost options
  root                Status and configuration of the root bridge
  summary             Summary of port states
  uplinkfast          Show spanning tree uplinkfast status
  vlan                VLAN Switch Spanning Trees
  |                   Output modifiers
  <cr>

#show spanning-tree mst ?
  <0-15>          MST instance id
  configuration   MST current region configuration
  detail          show detailed information
  interface       Spanning tree interface status and configuration
  |               Output modifiers
  <cr>
```

Also note the introduction of a detail keyword. Some few changes have been made to the spanning tree CLI in order to make it more efficient on switching platforms. See section 4.8 for more details on this.

4.1 New Global Configuration Commands Specific to MST

[no] spanning-tree mst forward-time *seconds*

[no] spanning-tree mst hello-time *seconds*

[no] spanning-tree mst max-age *seconds*

[no] spanning-tree mst max-hops *hopnumber*

[no] spanning-tree mst *[instance_id]* {root {primary | secondary} | priority *prio*}

4.1.1 spanning-tree mst forward-time

Syntax

[no] spanning-tree mst forward-time *seconds*

Description

This command changes the forward delay parameter for all the instances on the switch (this is an CIST instance parameter impacting all the MST instances).

seconds It must be a value in the range 4-30.

Example

(config)# spanning-tree mst forward-time 20

Default

Default value is 15 seconds.

4.1.2 mst hello-time

Syntax

[no] spanning-tree mst hello-time *seconds*

Description

This command changes the hello-time parameter for all the instances on the switch (this is a CIST instance parameter impacting all the MST instances).

seconds It must be a value in the range 1-10.

Example

(config)# spanning-tree mst hello-time 3

Default

Default value is 2 seconds.

4.1.3 spanning-tree mst max-age

Syntax

[no] spanning-tree mst max-age *seconds*

Description

This command changes the max-age parameter for all the instances on the switch (this is a CIST instance parameter impacting all the MST instances).

seconds It must be a value in the range 6-40.

Example

(config)# spanning-tree mst max-age 40

Default

Default value is 20 seconds.

4.1.4 spanning-tree mst max-hops

Syntax

[no] spanning-tree mst max-hops *value*

Description

This command changes the max-hop parameter for all the instances on the switch (this is a CIST instance parameter impacting all the MST instances). Inside an MST region, the MST instances don't use an age field in the messages they exchange. Instead, a hop count is included and incremented each time a switch relay such a message. When the hop count reaches the configured max-hops value, the information is discarded.

Setting the max-hops parameter can decrease the convergence time inside an MST region by reducing stale information that can circulate within the region before being discarded. The administrator can safely configure a value equal to the diameter of the region on the IST master and the backup IST master.

value An integer in the range 1-40 representing a number of possible hops in the region before a bpdu gets discarded.

Example

(config)# spanning-tree mst hello-time 3

Default

Default value is 20 hops.

4.1.5 spanning-tree mst priority

Syntax

[no] spanning-tree mst *instance_id* priority *prio*

Description

This command is used to change the priority of the bridge, which represents the likelihood that the bridge will be selected as the root bridge. Use the "no" form of this command to revert to the default value. MST always run in extended system id mode, which explains why the bridge priority can only be set by increment of 4096.

prio must be one of these numbers: 0, 4096, 8192, 12288, 16384, 20480, 24576, 28672, 32768, 36864, 40960, 45056, 49152, 53248, 57344, 61440.

instance_id Instance to which to apply the priority value. It must be a value in the range 0-15, 0 referring to the CIST instance.

<<Note on Extended System ID: Originally, a bridge ID consisted in a two byte priority field and a unique mac address. On a given switch, we used to allocate a different mac address for each vlan. Now the number of vlans is growing to 4096, this results in a waste of mac addresses. With extended system ID, we only allocated one mac address for a physical switch, and we take twelve bits from the priority field to create 4096 different bridge IDs. As there are only four high weight bits left for the effective priority, it can only be set by increment of 4096.>>

Example

(config)# spanning-tree mst 0 priority 4096

(config)# spanning-tree mst 2 priority 4096

Default

The default priority is 32768.

4.1.6 spanning-tree mst root

Syntax

[no] spanning-tree mst *instance_id* root {primary | secondary} [diameter *dia* [hello-time *hello*]]

Description

This command sets the priority and optionally the timer values for the bridge based on the network diameter. The **root secondary** set the priority value to 16384. The root primary tries to set a high enough priority (low value) to make the bridge root of the spanning tree instance. If the current root has the priority of 4096 then it fails, but the user can set the priority to 0 explicitly to make the switch root. The diameter option is only available for instance 0 (the CIST) because this option is in fact used to tune the spanning tree timers and that there is no timer configuration possible for the MST instances other than the CIST.

instance_id Instance to which to apply the priority value. It must be a value in the range 0-15.

Example

(config)# spanning-tree mst 0 root primary diameter 7 hello-time 2

(config)# spanning-tree mst 5 root primary

(config)# no spanning tree mst 3 root secondary

Default

The root command is a macro that possibly invoke several spanning tree parameter configuration commands. There is no default value but the no form of the command reverts to the defaults.

4.2 New Spanning Tree Global Configuration Commands

[no] spanning-tree portfast [bpdufilter | bpduguard] default

[no] spanning-tree loopguard default

[no] spanning-tree mode [pvst|mst]

4.2.1 spanning-tree loopguard default

Syntax

[no] spanning-tree loopguard default

Description

The loopguard feature provides an additional security in the bridge network by preventing alternate or root ports to become designated because of failure leading to a unidirectional link. This feature is more effective if configured in the bridged network as a whole. In order to facilitate its implementation, this command allows the user to enable loopguard as a default on all ports of a given bridge. Individual loopguard port configuration overrides this global default. See section 4.5.1. Loopguard is a feature that only operates on ports that are considered point-to-point by the spanning-tree. See section 4.5.5 to see exactly what a point-to-point link is.

<<Note on Loopguard: This feature especially works on ports that are receiving bpdus (whether blocking or root port) but is not harmful when configured on a designated port. If a loopguard-enabled port fails to receive bpdus that is was used to received from its designated peer, it will assume that a link failure occurred and break down, instead of going to forwarding. This broken state, called loop inconsistent, protects against unidirectional link failures and can be recovered from upon receipt of a new bpdu (Note that a loopguard blocking or root port can still go to designated forwarding as long as it receives inferior bpdus. It is only when it fails to receive bpdus that is goes loop inconsistent). See [9].>>.

Example

switch(config)# spanning-tree loopguard default

Default

Loopguard is disabled globally by default.

4.2.2 spanning-tree portfast default

Syntax

[no] spanning-tree portfast default

Description

RSTP introduces the concept of edge-port that is similar to Cisco portfast feature. In order to make the transition easier, the portfast keyword is used both for the global and the interface level configuration (section 4.5.2). This command is independent from the spanning tree mode used. On access switches, having a majority of access ports, we eventually want most of the switch's ports to end up configured as edge ports (portfast enabled). This command allows the user to control the default portfast configuration for all the ports of a given switch with a single command. The definition of edge port has a little bit evolved with 8021t support. We have now an operational portfast value that may be different from the portfast configuration: a portfast enabled port that receives a bpdu is not considered edge any more and so is not portfast enabled any more.

Example

switch(config)# spanning-tree portfast default

With this single line of command, all non-trunking switch ports that don't already have a specific portfast configuration will be treated like edge-ports (i.e. portfast enabled). The "no" form of the command reverts to the previous behavior (i.e. portfast disabled on all ports unless individually configured).

Default

The *no* form of the command is the default.

4.2.3 spanning-tree portfast bpdufilter default

Syntax

[no] spanning-tree portfast bpdufilter default

Description

This command enable bpdu filtering globally on the switch.

Bpdu filtering is a feature that prevent a port from sending or receiving any bpdus. As for the other "default" configuration commands, the effect of *portfast bpdufilter default* may be overridden by configuring bpdu filtering at the interface level (see 4.5.3). Tough, be careful that there is quite a difference in our implementation whether bpdu filtering is enabled on a per-port basis or globally.

When enabled globally:

- bpdufilter is only applied on ports that are in an **operational portfast** state.
- ports still send a few bpdus at link up before they effectively filter outbound bpdus.

Typically, the network administrator configures bpdu filtering globally on a switch so that ports leading to hosts are not disturbed by bpdus. This is why the feature is active on operational portfast port. If a bpdu is received on an edge port, as we saw in 4.2.2, it immediately loses it operational portfast status and thus bpdufilter is disabled. Also, because globally configured bpdufilter ports still send some few bpdus at link up, a misconfiguration where two such ports are connected together will not be catastrophic as the feature would immediately be disabled.

When enabled locally on a port, bpdufilter prevents the switch from receiving or sending bpdus on this ports with no exception. This is a potentially dangerous configuration that can create bridging loops if not correctly used, see 4.5.3.

Example

switch(config)# spanning-tree portfast bpdufilter default

switch(config)# no spanning-tree portfast bpdufilter

Default

The *no* form of the command is the default.

4.2.4 spanning-tree portfast bpduguard default

Syntax

[no] spanning-tree portfast bpduguard default

Description

Bpdu guard is a feature that basically disable a port if it ever receive a bpdu. This command enables globally bpdu guard on ports that are portfast enabled. This configuration is only applicable on ports that are in an operational portfast state and don't have an explicit interface bpduguard configuration (see 4.5.4). This will be useful on access switches where most ports are access (ports on which we generally don't want to accept bpdus).

This command was already existing, without the possibility of overriding it on a per-port basis. The previous syntax *spanning-tree portfast bpduguard* is still accepted but its new equivalent form *spanning-tree portfast bpduguard default* will be shown in the configuration.

Example

switch(config)# spanning-tree portfast bpduguard default

switch(config)# no spanning-tree portfast bpduguard

Default

The *no* form of the command is the default.

4.2.5 spanning-tree mode

Syntax

[no] spanning-tree mode [pvst | mst]

Description

Right now, a switch cannot run both pvst and mst (this may be possible later). The user can then switch between one mode to another using the spanning-tree mode command. The effect of changing mode are drastic: basically all the spanning-tree instances are stopped for the previous mode and restarted in the new mode. This implies of course disruption of user traffic; this command is to use with care.

Example

(config)# spanning-tree mode mst

(config)# spanning-tree mode pvst

(config)# no spanning-tree mode

Default

The default spanning tree mode is pvst. The no form of the command is equivalent to *spanning-tree mode pvst*.

4.3 MST Region Configuration Submode

A new submode for configuring the MST instance to vlan mapping is added to the global configuration mode. This mode is entered by the following global configuration command:

Syntax

[no] spanning-tree mst configuration

Description

This command enters the mst configuration submode. The no form of the command reset the mst configuration to its default.

Example

(config)# spanning-tree mst configuration

(config)# no spanning-tree mst configuration

Default

The mst configuration is made of 3 main parameters described below. The default value for the mst configuration is obtained using the default value for all its parameters:

- no vlan mapped to any mst instance (all vlans mapped to the CIST instance)
- name is an empty string.
- revision number is 0.

4.3.1 Instance-VLAN mapping

Syntax

[no] instance *instance_id* vlans *vlan-range*

no instance *instance_id*

Description

This *config-mst* submode command is used to map a vlan or a set of vlans to an MST instance. The mapping is incremental, not absolute. This means that when a range of vlans is provided, this one is added or removed to the existing ones.

Use the "no" form of this command to put the vlans back to the default instance (CIST).

instance_id instance to which the vlans *vlan-range* are mapped. It must be currently a value between 0 and 15. Any unmapped vlan is in fact mapped to the CIST instance so it is impossible to unmap a vlan from the CIST using the no form of the command.

vlan-range vlans to map to instance *instance_id*. It must be a value or a range in the range 1-4094.

Example

(config-mst)# instance 2 vlans 1-100

(config-mst)# no instance 2 vlans 40-60

(config-mst)# instance 10 vlan 10

(config-mst)# no instance 2

The first example adds vlan 1-100 to the ones already mapped (if any) to instance 2.

The second example moves back to the CIST instance all the vlans in the range 40-60 that were previously mapped to instance 2.

The third example adds vlan 10 to instance 10.

The fourth example removes all the vlans mapped to instance 2 and maps them back to the CIST instance.

Default

All the vlans that are not explicitly mapped an MST instance are mapped to the CIST instance, and by default there is no vlan mapping in the configuration (i.e. all the vlans are mapped to the CIST instance).

4.3.2 Region name

Syntax

[no] name *name*

Description

This *config-mst* submode command is used to set the configuration name. Two or more switches with the same vlan mapping and configuration version number are considered in different MST regions if the their configuration name are different. This parameter must be used with care as a typo would put the switch in a different region. Use the "no" version of this command to revert to the default region name. Be also aware that the standard defined the configuration name as a case sensitive parameter.

name Name to give the MST region. It can be any string with a maximum length of 32 characters.

Examples

(config-mst)# name Cisco

(config-mst)# name "Cisco Systems"

Default

The default value for the configuration name is an empty string "".

4.3.3 Configuration revision number

Syntax

[no] revision *version*

Description

This command is used to set the revision number of the mst configuration. Two switches with the same configuration but with different region version number are part of two different regions. This parameter must be treated with care.

The "no" version of this command sets the version value to its default.

version Configuration revision number, an integer in the range 0-65535.

Example

(config-mst)# revision 5

(config-mst)# no revision

Default

The default value for the mst configuration revision number is 0.

4.3.4 Exiting the submode with exit or abort

Syntax

abort
exit

Description

The abort and exit commands both allow the user to exit the mst configuration submode. To clearly understand the difference between these two commands, the way the configuration submode operates must be detailed further. As we already stated in section 3.1, the mst configuration is critical to the operation of the protocol: changing a single parameter in it can cause losses of connectivity. In order to reduce the number of service disruption, when the user enters the mst configuration submode, only a copy of the current mst configuration is edited. This way, when the user is done with editing the configuration, (s)he can apply all the changes at once using the **exit** keyword, or (s)he can exit the submode without committing any change to the configuration using the **abort** keyword.

We couldn't ensure exclusive access to the current mst configuration while it is being changed. This means that there is a slight chance that the changes could be lost instead of being applied when exiting the mst configuration submode. This should be very unlikely as it would only occur if two user commit a new configuration exactly at the same time. In this particular case, a warning message will be printed to the console:

```
Cosmos1(config-mst)#exit
% MST CFG:Configuration change lost because of concurrent access
```

A warning message related to private vlans synchronization can also be printed when exiting the mst configuration mode, see 4.3.6 for more information about that.

Example

(config-mst)# abort

(config-mst)# exit

Default

There is no default for these commands.

4.3.5 show commands

Syntax

show [current|pending]

Description

A weakness of our CLI is that the user has no possibility of checking the current configuration of the router while configuring it. (S)he has first to exit the configuration mode before being able to enter some show commands. This is not an option in the case of the mst configuration submode, where exiting means applying or discarding the configuration. We then had to introduce in the submode a minimum set of show commands to allow the user to verify the configuration (s)he is entering.

Show current will display the current configuration as it is used by the switch to run mst.

Show pending displays the edited configuration that will replace the current configuration if the user exit.

Example

```
(config-mst)#show pending
Pending MST configuration
Name      [zorglub]
Version   31415
Instance  Vlans Mapped
--------  ------------------------------------------------------------------
0         4001-4096
2         1010, 1020, 1030, 1040, 1050, 1060, 1070, 1080, 1090, 1100, 1110
          1120
3         1-1009, 1011-1019, 1021-1029, 1031-1039, 1041-1049, 1051-1059
          1061-1069, 1071-1079, 1081-1089, 1091-1099, 1101-1109, 1111-1119
          1121-4000
-------------------------------------------------------------------------------
```

A warning message related to private vlans synchronization can also be printed when displaying the mst configuration, see 4.3.6 for more information about that.

Default

There is no default value for these interactive commands.

4.3.6 Private vlans synchronization

Syntax

private-vlan synchronize

Description

It is highly desirable that primary and secondary private vlans share the same forwarding topology (see 7 for additional information on private vlans). In MST mode, different MST instance are typically created to have different topology, so a correct private vlan configuration will almost always lead to have primary and secondary vlans mapped to the same instance. As usual with IOS, this policy is not enforced and it is up to the user to create consistent private vlan to instance mapping. In order to help the administrator, a warning message is display when the mst configuration submode is exited and some secondary vlans are not mapped to the same instance as their primary. The mst configuration submode command private-vlan synchronize also allow the user to map all secondary vlans to the same instance as their primaries in one shot.

Example

The following example assumes a primary vlan 2 and a secondary vlan 3 associated to vlan 2.

```
#sh vlan private-vlan

Primary Secondary Type              Ports
------- --------- ----------------- ------------------------------------------
2       3         community
```

Assuming that all vlans were mapped to the CIST instance, let's change the mapping for the primary vlan 2 only:

```
#conf t
(config)#spanning-tree mst configuration
(config-mst)#instance 1 vlan 2
(config-mst)#exit
These secondary vlans are not mapped to the same instance as their primary:
```

```
-> 3
```

The warning message list the secondary vlans that are not mapped to the same instance as their primary. The command *show spanning-tree mst configuration* would warn the same way.

```
#show spanning-tree mst configuration
Name      []
Revision  0
Instance  Vlans mapped
--------  ---------------------------------------------------------------------
0         1,3-4094
1         2
---------------------------------------------------------------------------
These secondary vlans are not mapped to the same instance as their primary:
-> 3
```

Let's now use the private-vlan synchronize command in order to fix this:

```
#conf t
(config)#spanning-tree mst configuration
(config-mst)#private-vlan synchronize
(config-mst)#show pending
Pending MST configuration
Name      []
Revision  0
Instance  Vlans mapped
--------  ---------------------------------------------------------------------
0         1,4-4094
1         2-3
---------------------------------------------------------------------------

(config-mst)#
```

Of course, this example only shows a very simple MST configuration, the command will only be helpful when it can save a lot of manual configuration.

4.4 New Interface Submode Commands Specific to MST

[no] spanning-tree mst *instance_id* cost *cost*

[no] spanning-tree mst *instance_id* port-priority *prio*

The two first commands for the path cost and port-priority parameters are available in interface mode and allow to set these parameters for any MST instance (including the CIST with instance id 0).

```
(config)# interface g1/1
(config-if)#spanning-tree mst 1 ?
  cost           Change the interface spanning tree path cost for an instance
  port-priority  Change the spanning tree port priority for an instance
```

4.4.1 spanning-tree mst cost

Syntax

[no] spanning-tree mst *instance_id* cost *cost*

Description

This command is used to set the interface path cost. Use the "no" form of this command to revert to the default value. MST is always using the pathcost method long to compute the default cost of a port.

instance_id It must be a number in the range 0-15.

cost It must be a value in the range 1-200,000,000 Higher values indicate higher costs.

Example

(config-if)# spanning-tree mst 0 cost 17031970

(config-if)# spanning-tree mst 2 cost 2701

(config-if)# no spanning-tree mst 3 cost

Default

The default cost depends on the port speed: the faster is the interface speed and the smaller is the cost. MST always uses long pathcosts (see [8]).

4.4.2 spanning-tree mst port-priority

Syntax

[no] spanning-tree mst *instance_id* port-priority *prio*

Description

This command is used to set the interface priority. Higher values indicate smaller priorities.

instance_id It must be a number in the range 0-15 (remember that we will probably support a higher number of mst instance at FCS).

prio *A integer in the range 0-240 by increment of 16.*

<<Note: A spanning tree port ID is a 16 bit value split in two sections: port priority and port number. These two fields used to be 8 bit each, for a value in the range 0-255. Because switches can have more than 256 ports, 802.1t extended the port number field to 12 bits, only leaving 4 bits for the port priority. The port priority is still considered as a byte, but can only be set by increment of 16 in the range 0-240.>>

Example

(config)# interface FastEthernet 4/1

(config-if)# spanning-tree mst 0 port-priority 64

(config-if)# spanning-tree mst 2 port-priority 192

(config-if)# no spanning-tree port-priority

Default

The default value for the port priority is 128.

4.5 New Spanning Tree Interface Submode Commands

[no] spanning-tree guard [none | root | loop]

[no] spanning-tree portfast [disable | trunk]

[no] spanning-tree bpdufilter [enable | disable]

[no] spanning-tree bpduguard [enable | disable]

[no] spanning-tree link-type [point-to-point | shared]

4.5.1 spanning-tree guard

Syntax

[no] spanning-tree guard {none | root | loop}

Description

The spanning-tree guard interface command has been extended to support the loopguard feature along with the global-default configuration of loopguard. The previous syntax only allowed the user to enable root guard on a port using:

switch(config-if)# spanning-tree guard root

or disabling the feature, going back to the default:

switch(config-if)# spanning-tree guard none

The new syntax allow to explicitly enable loopguard or rootguard on the port, using the following syntax:

switch(config-if)# spanning-tree guard loop

or

switch(config-if)# spanning-tree guard root

The two features are exclusive. To disable both feature, the previous syntax has been kept:

switch(config-if)# spanning-tree guard none

The difference is that "none" is not the default value any more for spanning-tree guard, and hence would show up in the configuration if entered. The new default is the "no" form of the command:

switch(config-if)# no spanning-tree guard

In this case, root guard is disabled and loop guard is configured accordingly to the global-defaults (see section 4.2.1).

Example

switch(config-if)# spanning-tree guard loop

Default

The no version of the command is the default.

4.5.2 spanning-tree portfast

Syntax

spanning-tree portfast [disable | trunk]

no spanning-tree portfast

Description

Portfast is an existing command that has been enhanced in two ways:

- The feature can now operate on trunk ports, using the "trunk" keyword.
- Portfast can now have a default value globally set (see 4.2.2).

The following examples illustrate all the possibilities of the new version of portfast:

Example

switch(config-if)# spanning-tree portfast

This command enables portfast unconditionally on the given port.

switch(config-if)# spanning-tree portfast disable

This above configuration is disabling explicitly portfast for the given port. This is a change from the previous IOS releases. This configuration line will be showing up in the running-configuration as it is not the default.

switch(config-if)# spanning-tree portfast trunk

This new syntax allows the user to configure portfast on trunk ports. Note that with this configuration, the port will be configured for portfast even when in access mode.

switch(config-if)# no spanning-tree portfast

This command causes portfast to be enabled implicitly if the *spanning-tree portfast default* is defined in the global configuration and if the port is not a trunk port. This is the default and it is not shown in the configuration. If the user does not configure portfast globally, *no spanning-tree portfast* is equivalent to *spanning-tree portfast disable*. This will reduce confusion among customers who upgrade.

Default

switch(config-if)# no spanning-tree portfast

4.5.3 spanning-tree bpdufilter

Syntax

spanning-tree bpdufilter {enable | disable}

no spanning-tree bpdufilter

Description

Bpdu-filter allow the administrator to prevent a port from sending and receiving bpdus. The configuration is applicable to the whole interface, whether it is trunking or not. This configuration command has three states:

- enable: unconditionally enable bpdufilter on the interface
- disable: unconditionally disable bpdufilter on the interface
- no form: enable bpdufilter on the interface if it is in operational portfast state (see 4.2.2 & 4.5.2) and if the command spanning-tree portfast bpdufilter default is configured (see 4.2.3).

Be careful that enabling bpdufilter on an interface is approximately equivalent to disabling the spanning tree for this interface. It is definitely possible to create bridging loops when this command is not correctly used.

Example

switch(config-if)# spanning-tree bpdufilter enable

Enables bpdu filtering on the interface.

switch(config-if)# no spanning-tree bpdufilter

Enable bpdufilter on the interface if it is configured globally.

Default

switch(config-if)# no spanning-tree portfast bpdufilter

4.5.4 spanning-tree portfast bpduguard

Syntax

spanning-tree bpduguard {enable | disable}

no spanning-tree bpduguard

Description

Bpduguard is a feature that prevents a port from receiving bpdus. This is typically used in a service provider environment where the administrator wants to prevent an access port from participating in the spanning tree. If the port still receives a bpdu, it is put in the ErrDisable state, as a protective measure.

This configuration command has three states:

- enable: unconditionally enable bpduguard on the interface
- disable: unconditionally disable bpduguard on the interface
- no form: enable bpduguard on the interface if it is operational portfast state (see 4.2.2 & 4.5.2) and if the command spanning-tree portfast bpduguard default is configured (see 4.2.4).

Example

switch(config-if)# spanning-tree bpduguard enable

Allows a port to receive bpdus.

switch(config-if)# no spanning-tree bpduguard

Accept bpdus if bpduguard is not globally enabled and port in portfast mode.

Default

switch(config-if)# no spanning-tree bpduguard

4.5.5 spanning-tree link-type

Syntax

spanning-tree link-type {point-to-point | shared}

no spanning-tree link-type

Description

With RSTP, a port with a designated role can transition directly to the forwarding state if it is connected to a peer port also understanding RSTP. In order to achieve this fast convergence, the designated port must negotiate its rapid transition with a unique neighboring bridge and this implies that the link must be known by the protocol as a point to point link (see [1] and [4]).

By default, the switch will derive the link type of a port from the duplex mode: a full-duplex port will be considered as point-to-point while a half-duplex configuration will be assumed to be on a shared link. Of course, this automatic setting may need to be overridden: a half-duplex link can still be physically connected point-to-point to a single port on a remote bridge running RSTP, thus allowing rapid transition. More seldom would be the case where a full-duplex link couldn't be considered as a point-to-point link - though, it is still possible at the spanning-tree protocol level, indroducing switches transparent for this protocol. The command mst link-type allows the user to configure a link type for a port.

Example

switch(config-if)# spanning-tree link-type shared

This port will be considered as a shared link, forbidding RSTP fast transition, regardless of the duplex setting.

switch(config-if)# no spanning-tree link-type

The link-type is guessed from the duplex mode as a result of the above configuration.

Default

The no form of the command will be the default: i.e. link-type will be automatically derived from the duplex operational setting unless explicitly configured.

4.6 New Exec Commands

4.6.1 clear spanning-tree detected protocol

Our new implementation of MST will introduce a new enable exec command:

syntax:

switch# clear spanning-tree detected-protocol [interface]

Description:

RSTP and MST have some built-in compatibility mechanisms that allow them to interact properly with other version of IEEE spanning-tree or other regions. For instance, a bridge running RSTP is able to send 802.1D BPDUs on one of its ports when it is connected to a legacy bridge. A MST bridge is also able to detect that a port is at the boundary of a region when it receives a legacy BPDU or a MST BPDU associated with a different region. These mechanisms are unfortunately not always able to revert to the most efficient mode. A RSTP bridge designated for a legacy 802.1D will stay in 802.1D mode even after the legacy bridge has been removed from the link for instance. The same way, a MST port can still assume it is a boundary port when the bridge(s) to which it is connected have joined the same region. In order to force renegotiation with the neighbors, the command *clear spanning-tree detected-protocol* needs to be entered by the user.

Example:

If the user specify an interface, the command will only be applied to this interface:

switch# clear spanning-tree detected-protocol fa1/1

else, the command will be applied to each and every port of the switch:

switch# clear spanning-tree detected-protocol

Default:

There is no default for this interactive command.

4.6.2 clear spanning-tree counters

This command have been requested for a while. It allows the user to clear the BPDU count on the interfaces and to clear the uplinkfast/backbonefast statistics.

Syntax

clear spanning-tree counters
clear spanning-tree counters interface [X]

This first form of the command clear the uplinkfast/backbonefast statistics and the BPDU count on every ports.

The second form only clears the BPDU count on the specified interface but leave the uplinkfast/backbonefast statistics unchanged.

Example

switch# clear spanning-tree counters

Notes

- We initially thought of linking this command to the clear counters exec command. Problem is that clear counters is only an interface command. When the user enters clear counters without specifying an interface, the command is just replicated on each and every port on the switch. This means that we have no way of determining when the clear counters has been entered for a specific interface or for all interfaces. Thus, we would not have been able to clear the uplinkfast/backbonefast because we wouldn't have been able to detect when the user wants to clear all the statistics instead of just clearing the BPDU count on a port.
- SNMP counters are not affected by this command.

4.7 New Show Commands Specific to MST

show spanning-tree mst configuration

show spanning-tree mst [instance_id] [detail]

show spanning-tree mst [instance_id] interface *interface* [detail]

4.7.1 show spanning-tree mst configuration

Syntax

show spanning-tree mst configuration

Description

This command is used to print the region configuration, which consists in three main elements: vlan mapping (vlan to instance), region name and configuration version. A warning message related to private vlans synchronization can also be printed when displaying the mst configuration. see 4.3.6 for more information about that.

Example

```
#show spanning-tree mst configuration
Name      [leo]
Revision  2702
Instance  Vlans mapped
--------  ---------------------------------------------------------------------
0         1-9,11-19,21-29,31-39,41-4094
1         10,20,30,40
```

4.7.2 show spanning-tree mst

Syntax

show spanning-tree mst [instance id] [detail]

Description

It was necessary to introduce such a command to display the additional values specific to the MST protocol. By default, when no instance is explicitly specified by the user, all instances currently running on the switch will be displayed, the one after the other. When in MST mode, there is a least MST00 that is running.

Example

The following example displays MSTP information for all MST instances currently running:

```
#show spanning-tree mst

###### MST00        vlans mapped:   1-2,4-2999,4000-4094
Bridge      address 0002.172c.f400  priority  49152 (49152 sysid 0)
Root        address 0050.3e66.d000  priority  8193  (8192 sysid 1)
            port    Gi1/1           path cost 20004
IST master  this switch
Operational hello time 1, forward delay 15, max age 20
Configured  hello time 2, forward delay 15, max age 20, max hops 35

Interface        Role Sts Cost      Prio.Nbr Status
---------------- ---- --- --------- -------- --------------------------------
Gi1/1            Root FWD 20000     128.1    P2p Bound(PVST)
Fa4/1            Desg FWD 200000    128.193  P2p
Fa4/2            Back BLK 200000    128.194  P2p

###### MST03        vlans mapped:   3,3000-3999
Bridge      address 0002.172c.f400  priority  32771 (32768 sysid 3)
Root        this switch for MST03

Interface        Role Sts Cost      Prio.Nbr Status
---------------- ---- --- --------- -------- --------------------------------
Gi1/1            Boun FWD 20000     128.1    P2p Bound(PVST)
Fa4/1            Desg FWD 200000    128.193  P2p
Fa4/2            Back BLK 200000    128.194  P2p
```

See section 4.8.3 for the list of possible values displayed under the Status column.

The output can be restricted to a given instance by specifying an instance id after the mst keyword. Some more detailed information can be printed by happending the **detail** keyword at the end of the command line. The following example shows detailed output for instance 3:

```
#show spanning-tree mst 3 detail

###### MST03        vlans mapped:   3,3000-3999
Bridge      address 0002.172c.f400  priority  32771 (32768 sysid 3)
Root        this switch for MST03

GigabitEthernet1/1 of MST03 is boundary forwarding
Port info             port id           128.1  priority    128  cost       20000
Designated root       address 0002.172c.f400  priority  32771  cost           0
Designated bridge     address 0002.172c.f400  priority  32771  port id    128.1
Timers: message expires in 0 sec, forward delay 0, forward transitions 1
Bpdus (MRecords) sent 4, received 0

FastEthernet4/1 of MST03 is designated forwarding
Port info             port id           128.193  priority    128  cost       200000
Designated root       address 0002.172c.f400  priority  32771  cost           0
Designated bridge     address 0002.172c.f400  priority  32771  port id    128.193
Timers: message expires in 0 sec, forward delay 0, forward transitions 1
Bpdus (MRecords) sent 254, received 1

FastEthernet4/2 of MST03 is backup blocking
Port info             port id           128.194  priority    128  cost       200000
Designated root       address 0002.172c.f400  priority  32771  cost           0
Designated bridge     address 0002.172c.f400  priority  32771  port id    128.193
Timers: message expires in 2 sec, forward delay 0, forward transitions 1
Bpdus (MRecords) sent 3, received 252
```

4.7.3 show spanning-tree mst interface

Syntax

show spanning-tree mst [*instance id*] interface *interface* [detail]

Description

The user can refine the output of the show spanning-tree mst command by specifying an interface.

Example

The following example shows the mst information for all mst instances on interface FastEthernet4/1:

```
#show spanning-tree mst interface fastEthernet 4/1

FastEthernet4/1 of MST00 is designated forwarding
Edge port: no             (trunk)          port guard : none        (default)
Link type: point-to-point (point-to-point) bpdu filter: disable     (default)
Boundary : internal                        bpdu guard : disable     (default)
Bpdus sent 349, received 3

Instance Role Sts Cost      Prio.Nbr Vlans mapped
-------- ---- --- --------- -------- --------------------------------
0        Desg FWD 200000    128.193  1-2,4-2999,4000-4094
3        Desg FWD 200000    128.193  3,3000-3999
```

Same example, detail version and only specifying information for instance 0:

```
#show spanning-tree mst 0 interface fastEthernet 4/1 detail
Edge port: no             (trunk)          port guard : none        (default)
Link type: point-to-point (point-to-point) bpdu filter: disable     (default)
Boundary : internal                        bpdu guard : disable     (default)

FastEthernet4/1 of MST00 is designated forwarding
Vlans mapped to MST00 1-2,4-2999,4000-4094
Port info             port id        128.193  priority    128  cost      200000
Designated root       address 0050.3e66.d000  priority   8193  cost       20004
Designated ist master address 0002.172c.f400  priority  49152  cost           0
Designated bridge     address 0002.172c.f400  priority  49152  port id  128.193
Timers: message expires in 0 sec, forward delay 0, forward transitions 1
Bpdus sent 492, received 3
```

4.8 Other Show Command Enhancements

4.8.1 The "brief" keyword replaced by a "detail" keyword

As seen in the previous chapters, the MST show commands display a summary by default and only provide additional information when the *detail* keyword is used. The behavior is closer to catos implementation and is definitely more efficient on switching platform that can potentially host several hundreds of ports.

Users who created scripts using the spanning tree show commands have to remove the *brief* keyword if they were using it, or add the *detail* keyword if they were not using *brief*.

Show commands affected by this change are:

- show spanning-tree brief
- show spanning-tree active brief (Note: the *active* keyword in not interpreted anyway)
- show spanning-tree vlan *vlanid* brief
- show spanning-tree [vlan *vlanid*] interface *interface* brief
- show spanning-tree root brief

4.8.2 The "wide" keyword disappears

Initialy supposed to display information on a 132 column terminal, the wide keyword has never been used by the spanning-tree show commands, even though it was configurable by CLI.

4.8.3 The port summary supports long cost format and is PVRST ready!

With the introduction of long cost, some show commands were not displayed correctly on a 80 character terminal (see [8]). We removed a redundant column in the following output in order to accomodate the long costs. We also took the opportunity to slightly change the output of these formerly "brief" commands.

```
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    4097
              Address     0002.172c.f400
              This bridge is the root
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    4097   (priority 4096 sys-id-ext 1)
              Address     0002.172c.f400
              Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
```

```
                Aging Time 300

Interface        Role Sts Cost      Prio.Nbr Status
---------------- ---- --- --------- -------- --------------------------------
Gi1/1            Desg FWD 20000     128.1    P2p
Fa4/1            Desg FWD 200000    128.193  Edge P2p
Fa4/2            Desg FWD 200000    128.194  Edge P2p
```

Changes:

- You can notice on the above example that the column priority has been removed. That was superfluous as, anyway, the port priority is part of the port id. The Prio.Nbr column now is the only place where is displayed the priority and the port number.
- The interface name is now displayed using its short format.
- A Role column has been added. 802.1D did not define a role, but we use the current 802.1w definition to give additional information to the user. Possible values are **Desg** (designated), **Root**, **Altn** (alternate) and **Back** (backup).
- The designated bridge information has now been replaced by a Status column. The designated bridge information is only really useful during precise troubleshooting and was anyway not understood by most users. The information is still available in the detail form of the show command.
- Here is a summary of what can be displayed under the Status column:

P2p/Shr: the interface is considered as a point-to-point(resp. shared) interface by the spanning-tree.

Edge: the port has an oper-portfast status. This means that portfast has been configured (whether globally via the default command or directly on the interface) and that no BPDU has been received.

ROOT_Inc, *LOOP_Inc, *PVID_Inc** and ***TYPE_Inc**: this status will be shown when the port is in a broken state (**BKN) for an inconsistency. The port would be respectively Root inconsistent, Loopguard inconsistent, PVID inconsistent or Type inconsistent.

Bound(type): When in MST mode, this keyword will identify boundary ports, specifying the type of the neighbor (STP, RSTP, PVST, PVRST).

Peer(STP): In PVRST mode, this keyword will be display to identify a port connected to a legacy 802.1D bridge.

This few changes make the output of the show spanning-tree vlan command more concise and introduce all the new fields that will be necessary for future support of PVRST (Per Vlan Rapid Spanning Tree).

4.8.4 Show spanning-tree vlan is still supported in MST mode

Even when the switch is configured to run in mst mode, the show spanning-tree vlan commands are accepted. The information that is then displayed is concerning the MST instance to which the specified vlan is mapped.

For instance, let`s suppose that vlan 123 is mapped to instance 0:

```
cosmos2#show spanning-tree vlan 123

MST00
  Spanning tree enabled protocol mstp
  Root ID     Priority    8193
              Address     0050.3e66.d000
              Cost        20004
              Port        1 (GigabitEthernet1/1)
              Hello Time   1 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    49152  (priority 49152 sys-id-ext 0)
```

```
                Address     0002.172c.f400
                Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

Interface        Role Sts Cost      Prio.Nbr Status
---------------- ---- --- --------- -------- --------------------------------
Gi1/1            Root FWD 20000     128.1    P2p Bound(PVST)
Fa4/1            Desg LRN 200000    128.193  P2p
Fa4/2            Back BLK 200000    128.194  P2p
```

4.8.5 Show spanning-tree summary displays new features' status

Several new features have been introduced with this release. The show spanning-tree summary command will list the status for all of them. Notice also that the commands displays the spanning-tree instances for which this switch is root in a range format. This should greatly reduce the length of the output and increase its readability:

example:

```
#show spanning-tree summary totals
Root bridge for: Bridge group 1, VLAN0001, VLAN0004-VLAN1005
  VLAN1013-VLAN1499, VLAN2001-VLAN4094
EtherChannel misconfiguration guard is enabled
Extended system ID   is enabled
Portfast             is enabled by default
PortFast BPDU Guard  is disabled by default
Portfast BPDU Filter is disabled by default
Loopguard            is disabled by default
UplinkFast           is disabled
BackboneFast         is disabled
Pathcost method used is long

Name                   Blocking Listening Learning Forwarding STP Active
---------------------- -------- --------- -------- ---------- ----------
1 bridge                  0        0         0        1          1
3584 vlans                3584     0         0        7168       10752

                       Blocking Listening Learning Forwarding STP Active
---------------------- -------- --------- -------- ---------- ----------
Total                     3584     0         0        7169       10753
```

5.0 SNMP

The Bridge MIB working group of IETF is likely to bring out an RFC for 802.1s MIB. However efforts are underway to provide SNMP support using CISCO-STP-EXTENSIONS-MIB. The STP core and MST implementation shall provide the API necessary for the MIB implementation. <TBD>

6.0 References

[1] "IEEE 802.1w Rapid Spanning Tree", IEEE 802.1 Interworking Task Group

[2] "IEEE 802.1s Multiple Spanning Trees", IEEE 802.1 Interworking Task Group

[3] "802.1w Rapid STP and 802.1s Multiple STP Architecture Spec.", Shyam Kaluve, ENG-90247

[4] "802.1w Rapid STP and 802.1s Multiple STP Functional Spec", Shyam Kaluve, ENG-90248

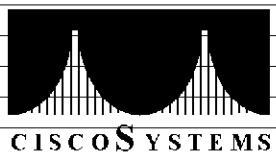
[5] "IOS Multi Instance Spanning Tree Protocol Functional Spec", Bill Rainey, ENG-56419

[6] "CatOS 802.1s Functional/Design Spec", Prabhu Seshachellam, et. al. ENG-102657

[7] "Private VLANs Under IOS", Frederick Scott & Dan Florea, ENG-56211

~~[8] "STP 32 bit Default pathcost in IOS", Bill Rainey & Srikanth Kilaru, ENG-72415~~

[9] "STP Loop Guard", Shyam Kaluve, ENG-104341



• May 17th 2001 meeting minutes

- We did not came to a final decision about the implementation of 1s on routers (with bridge groups). Would it make sense? RSTP alone would be easy, 1s could introduce a few problems. NOTE: Whatever we will decide will apply to both Native and Hybrid.
- Single edit buffer: Iris confirmed that each session can attach its private buffer to the csb and so no lock is needed (unlike CatOS).

- [no] spanning-tree global-default loopguard

COMMENTS:

- If you do not specify the interface, it is already a global command and so we should not need the "global default command
- "default" instead of "global-default" is a bad choice too because every IOS command can be started already with the "default" keyword
  ex. #default spanningtree vlan 2 ....
  Even if it is used very little, it exists ...
- The "range" keyword would help us but not completely: it does not handle modules inserted or removed after the command has been typed
- Being a new feature for IOS, it should be disabled by default (What would it happen if released with bugs?)

WE SUGGESTED:

- [no] spanning-tree loopguard-default
  In this way it can be used by PVST as well.
- [no] spanning-tree global-default edge-ports

COMMENTS:

- Same problem as with loopguard for the keyword "global-default"
- It does not make much sense to use the keyword "edge-ports" for the global configuration and "portfast" for the interface configuration: it is better to be consistent using the same one in both the cases. Moreover, if there is a global edge-port there should be a non global edge-port!

WE SUGGESTED:

- [no] spanning-tree portfast-default
  Since the standard committee would like to have the "edge-port" keyword, we can add it to the help associated to the command "portfast-default". In this way we can keep the "portfast" keyword and underline that it means "edge-port". In this way it can be used by PVST as well. It was also proposed to use a sort of "implicit" default value for the "edge-port" flag and to apply it when entering in mst mode.

ACCESS SW vs BACKBONE SW

The problem of this approach is that ACCESS switches and BACKBONE switches would like two different defaults. The number of ACCESS switches being larger than the others it would make sense to go for a "edge-port" as default.

POST MEETING:

- It could be possible to have both the commands (portfast and edge-port) with the same meaning and nvgen only one. This solution has two drawbacks:
- It is never a good idea to have two different commands doing the same thing.
- It could confuse the user: the user could configure the feature using the "edge-port" keyword and later see the "portfast" keyword in the config file.

SUBMODE FOR THE REGION CONFIGURATION

We can not have a submode without a way to see the differences between the old and the new configuration before to apply them. A sort of "discard" is needed as well. We decided for the introduction of the following three commands inside the submode:

- exit (apply the changes)
- show (shows the differencies between the old and the new configuration)
- abort (discard the changes)

NOTE: We had already this model in mind earlier, but we reverted to the one without those new commands because we saw that all of them had been removed from the vlan database submode. We thought that if they did that it was because of a good reason.

ALTERNATIVES:

It was proposed to introduce a sort of

- #show edit-buffer

that would have done the same as the show command we are suggesting to put inside the submode. An "exec" command, which allows you to execute any command from any submode, would solve the problem, but at the moment we are not aware of any concrete plan for the introduction of it. It is true that the IOS CLI infrastructure allows us do all we want: we have just to associate a specific function (that we write and thus we control) to the new token, but the introduction of those 3commands would arise two contradictions:

What we add is what has been recently removed from the vlan database submode (There may be a good reason for that). It is not normal in IOS to have the buffer concept even inside a submode

POST MEETING:

It has been pointed out by Bill Rainey that the "show" and "discard" commands do not fit well in the IOS config mode model. Another possible solution is the following: We allow the user do have more than one mapping and decide which one to apply. In this way it would be possible to enter in config mode, edit the mapping #2, exit from config mode, do something like

- show spanning-tree mst region-config mapping 2 (this is just an example)

and if the mapping is what we want we can apply it, otherwise we can go back in config mode and fix it. This would also allow us to import the vlan mapping[s] from a tftp-server, making it/them unique and easier to distribute among the switches in the MST domain.

PORTFAST SYNTAX

- spanning-tree portfast [disable | trunk]
- [no] spanning-tree portfast

It was suggested to remove the "trunk" keyword: if before we did not honor the portfast flag for trunk ports, now we simply do! That keyword is just superfluous. Unfortunately it is not possible for backward compatibility reasons: the old model is such that the same interface honors the flag if it is not trunk and it does not honor it otherwise. If we do not respect that condition we would break the old configurations because where trunk ports were supposed not to honor it, they would.

LOOPGUARD/ROOTGUARD

#[no] spanning-tree guard [none | root | loop]

We think that the "none" keyword did not have much sense before (it simply becomes an alternative to the no command) but it has one now. The reason of the none keyword is "look at the global default"

RSTP

spanning-tree mst redetect-neighbors [interface]

This command is used to restart the algorithm that decided what type of "Neighbors" we are talking to: Rapid spanning tree capable or not. We decided that "neighbors" was a bad keyword, for two main reasons:

-There are not real neighbors, what would happen if the user tried to do something like this:

#show spanning-tree neighbors

The same keyword is already used in many other contexts, ranging from CDP to the different routing protocols. We decided for the new keyword "redetect-protocol"

POST MEETING:

Bill Rainey suggested the use of the "clear" command. We could have something like:

#clear spanning-tree mst neighbors

Another alternative could be:

#spanning-tree mst restart-migration

• July 23rd 2001 first parser-police discussion

Introduced the CLI for the mst configuration submode (detailing the show commands, abort exit etc...).

The show commands have been renamed as they are in the latest release of the document. The vlan to instance mapping syntax has been greatly simplified.

• September 26th 2001, second parser-police discussion

Proposed the global default commands. There has not been many changes following this second parser police posting. The dashes (loopguard-default) have been dropped, this is the main change.